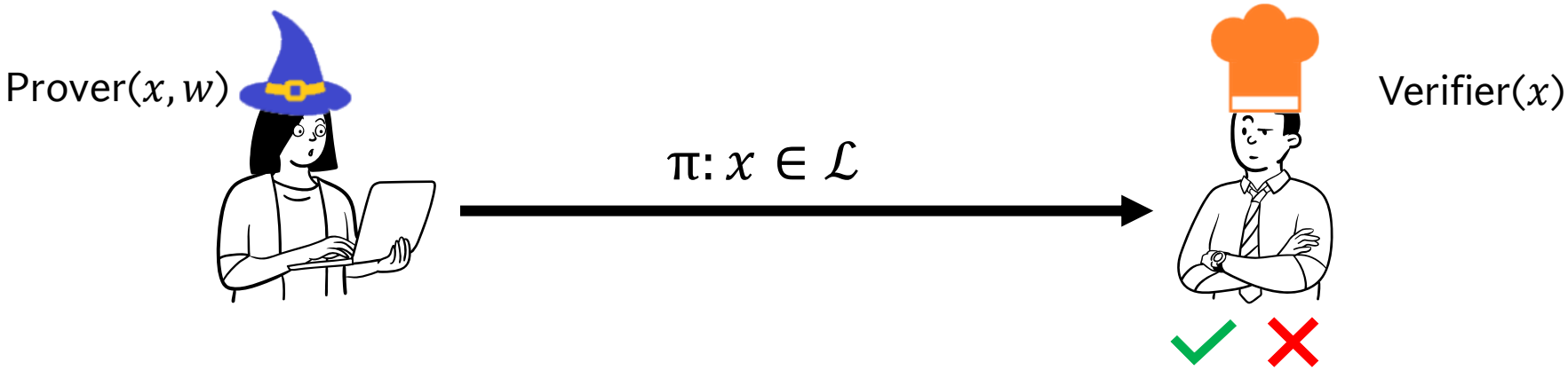
Non-Interactive Zero-Knowledge From Non-Interactive Batch Arguments

Jeffrey Champion and David Wu

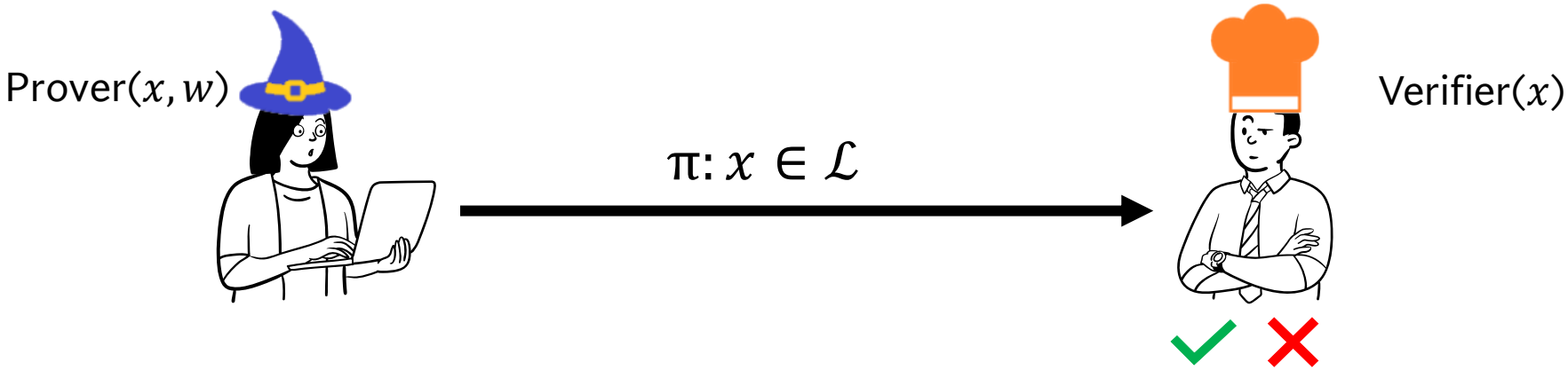
Non-interactive Proof for $\mathcal{L} \in$ NP



Completeness: “honest proofs verify”

Soundness: “no (efficient) prover can produce π that verifies with $x \notin \mathcal{L}$”

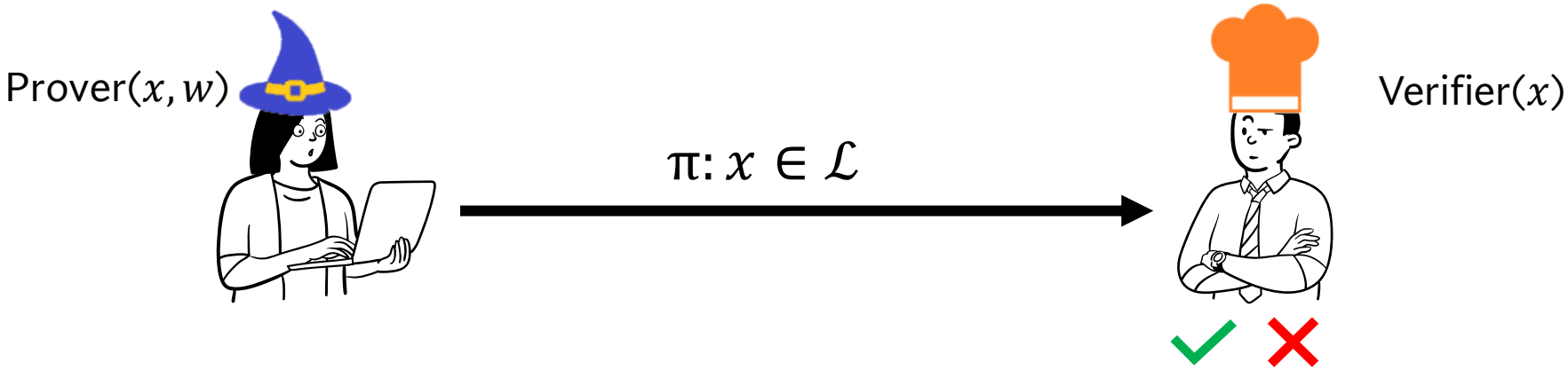
Non-interactive Proof for $\mathcal{L} \in$ NP



Basic Questions:

- How short can π be?
- Can π hide information on w?

Non-interactive Proof for $\mathcal{L} \in \mathrm{NP}$



Basic Questions:

- How short can π be? Motivates "succinctness"
- Can π hide information on w? Motivates "zero-knowledge"

Recent work: [GOS12], [BCCT12], [DFH12], [Lip13], [PHGR13], [GGPR13], [BCI+13], [BCPR14], [Gro16], [BISW17], [BCC+17], [BISW18], [BBHR18], [CCH+19], [PS19], [LPWW20], [BKM20], [COS20], [CHM+20], [Set20], [JJ21], [ACL+22], [BS23], [CBBZ23]

Intuitively, a short enough proof should lose information about w...

Can we show that succinctness implies zero-knowledge?

Previous work [KMY20]

Succinct non-interactive argument where proof length < witness length

- OWFs + SNARG for NP ⇒ computational NIZK argument for NP

Previous work [KMY20]

Succinct non-interactive argument where proof length < witness length

- OWFs + SNARG for NP ⇒ computational NIZK argument for NP

SNARGs for NP with adaptive soundness require strong, non-falsifiable assumptions [GW11]

Can we relax succinctness and get an analogous result?

Batch Arguments for $\mathcal{L} \in$ NP [KVZ21], [CJJ21]

Prover$(x_1, \ldots, x_m, w_1, \ldots, w_m)$

Verifier$(x_1, \ldots, x_m)$

π: $x_1, \ldots, x_m \in \mathcal{L}$

Amortized over # of statements

Succinctness: $|\pi| = o(m) * (|x| + |w|)$

Why should we care about BARGs for NP?

- A lot of very recent work: [KVZ21], [CJJ21], [HJKS22], [WW22], [DGKV22], [GSWW22], [CGJ+22], [KLVW23]
- Batch languages can be viewed as a special subset of NP where we can get SNARGs from falsifiable assumptions despite Gentry-Wichs
- Succinct proofs of batch languages can still be useful in cases where a SNARG would be (e.g. aggregate signatures)
- BARGs for NP can also be used to get SNARGs for P [KVZ21], [CJJ21]

Note: BARG + NIZK ⇒ zkBARG *(requires certain properties)

This work

Each output bit depends on a small number of input bits

Sub-exp secure local PRG

+

(Somewhere sound) BARG for NP

+

Lossy PKE (dual-mode commitment)

⇒

NIZK argument for NP with
- Computational ZK
- Adaptive soundness

Two modes: statistically hiding and statistically binding

Proofs in the Hidden-Bits Model [FLS90]

$r =$ | r_1 | r_2 | r_3 | r_4 | r_5 | r_6 | … | r_m |

Prover(x, w)

$\pi, I \subseteq [m]$

Verifier(x)

Prover has access to a uniform string r

Proofs in the Hidden-Bits Model [FLS90]

$r =$ | r_1 | r_2 | r_3 | r_4 | r_5 | r_6 | ... | r_m |

Prover(x, w)

Prover has access to a uniform string r

$\pi, I \subseteq [m]$

Verifier(x)

Verifier only sees the subset of bits in I

Zero-knowledge: verifier does not see bits in $[m] \setminus I$

Soundness: prover has no control over the string r

We can construct NIZKs in the HBM unconditionally!

From HBM NIZK to NIZK

$r =$ | r_1 | r_2 | r_3 | r_4 | r_5 | r_6 | … | r_m |

Prover(x, w)

$\pi_x, I \subseteq [m]$

Verifier(x)

From HBM NIZK to NIZK

CRS

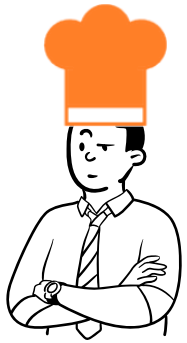
Prover(x, w)

$\pi_x, I \subseteq [m]$

$r = (r_1, \ldots, r_m)$

π_I, r_I

Verifier(x)

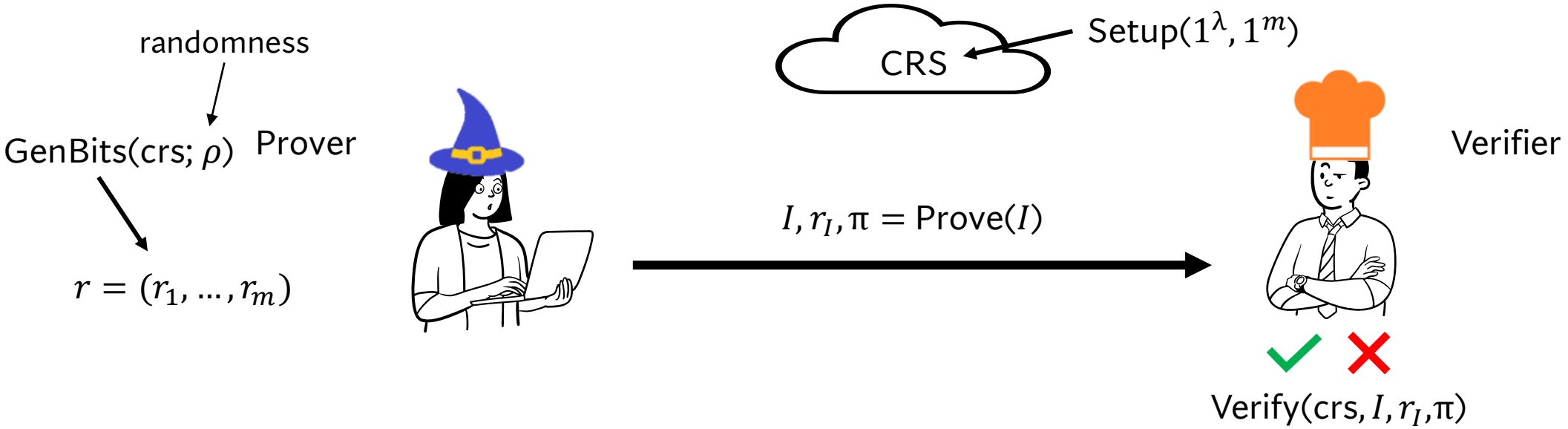


Verifier checks both proofs are valid

Prover samples m bits from a sparse subset of $\{0,1\}^m$ with randomness ρ such that $|\rho| < m$

proof that bits were “correctly” sampled

Hidden-Bits Generators [QRW19], [KMY20]



Binding: $\exists \mathcal{V}^{\text{crs}} \subseteq \{0,1\}^m$ such that:
(1) $\mathcal{V}^{\text{crs}}$ is a **sparse** subset
(2) No PPT prover can output a valid proof where $r_I \notin \mathcal{V}_I^{\text{crs}}$

Hiding: No PPT verifier can distinguish $r_{\bar{I}}$ from uniform given (crs, I, r_I, π) sampled honestly

Note: HBG => NIZK

Warm-up: Constructing HBGs

Sketch of [KMY20]:

- **Hidden-bits string:** $r = \mathrm{PRG}(s)$ for random s
- **Proof:** a SNARG proof that $\exists s$ s.t. $\forall i \in I\colon r_i = \mathrm{PRG}(s)_i$

batch language

- **Binding:**
  - r's form sparse subset of $\{0,1\}^m$ (PRG expansion)
  - No PPT prover can output valid π that disagrees with this set (SNARG soundness)
- **Hiding:**
  - $r_{\bar{I}}$ is close to uniform given $(\mathrm{crs}, I, r_I, \pi)$ sampled honestly (SNARG proof is short enough to argue leakage resilience)

What happens if we switch SNARG to BARG?

- BARG proof is longer than the PRG seed
- Each witness could be different seed

Warm-up: Constructing HBGs

Sketch of [KMY20]:

- **Hidden-bits string:** $r = \mathrm{PRG}(s)$ for random s
- **Proof:** a SNARG proof that $\exists s$ s.t. $\forall i \in I\colon r_i = \mathrm{PRG}(s)_i$ batch language
- **Binding:**
  - r's form sparse subset of $\{0,1\}^m$ (PRG expansion)
  - No PPT prover can output valid π that disagrees with this set (SNARG soundness)
- **Hiding:**
  - $r_{\bar{I}}$ is close to uniform given $(\mathrm{crs}, I, r_I, \pi)$ sampled honestly (SNARG proof is short enough to argue leakage resilience)

What happens if we switch SNARG to BARG?

- BARG proof is longer than the PRG seed Fix: local PRG to shorten witness size
- Each witness could be different seed Fix: commit to PRG seed and prove consistency

HBG Construction

Setup($1^\lambda, 1^m$)

crs = (crsBARG, crsBIND)

Note: extractor seeds also in crs

Prover

GenBits(crs; ρ):

random seed

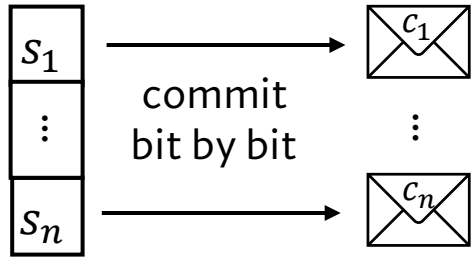
PRG(s_1 s_2 s_3 s_4 ... s_n)

$t =$ t_1 t_2 t_3 ... t_{mB}

break t into blocks: $B_1 = t_1 \dots t_B$ ... B_m

Ext(B_1) = hidden bits r_1 ... r_m = Ext(B_m)

Prove(I):

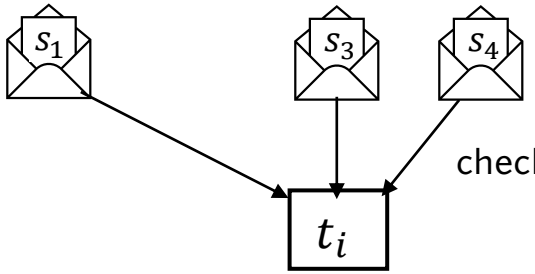


Verifier

Verify(crs, I, r_I, π): check extraction and BARG proof

Proving BARG statement $i \in [mB]$:

consider local indices



open commitments

check PRG output matches

$\pi = (\pi_{BARG}(I), B_I, c_1, \dots, c_n)$

Proof: Binding

Binding: $\exists \mathcal{V}^{\text{crs}} \subseteq \{0,1\}^m$ such that:
(1) $\mathcal{V}^{\text{crs}}$ is a sparse subset
(2) No PPT prover can output a valid proof where $r_I \notin \mathcal{V}_I^{\text{crs}}$

Setup$(1^\lambda, 1^m)$

crs = (crsBARG, crsBIND)

Prover

GenBits(crs; ρ):

random seed

PRG(s_1 s_2 s_3 s_4 ... s_n)

$t =$ t_1 t_2 t_3 ... t_{mB}

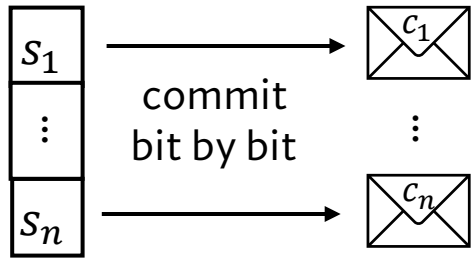
break t into blocks: $B_1 = t_1 \ldots t_B$ ... B_m

Ext(B_1) = hidden bits r_1 ... r_m

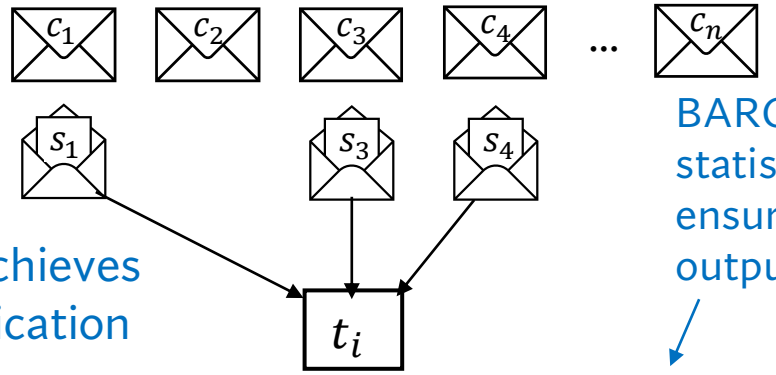
Ext(B_m) = hidden bits r_1 ... r_m

PRG achieves sparsification

Prove(I):



Proving BARG statement $i \in [mB]$:



BARG soundness + statistical binding ensure that only PRG outputs are valid

$\pi = (\pi_{BARG}(I), B_I, c_1, \ldots, c_n)$

Note: full details are in the paper

Hiding: No PPT verifier can distinguish $r_{\bar{I}}$ from a uniform bitstring given (crs, I, r_I, π) sampled honestly

Proof: Hiding

Setup($1^\lambda, 1^m$) → crs = (crsBARG, crsBIND)

Prover

GenBits(crs; ρ):

random seed

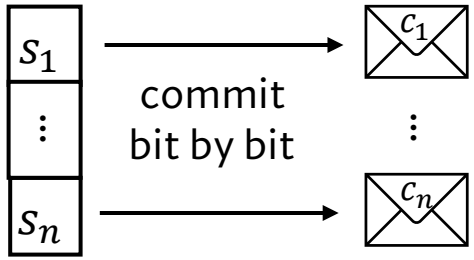
PRG(s_1 s_2 s_3 s_4 … s_n)

$t =$ t_1 t_2 t_3 … t_{mB}

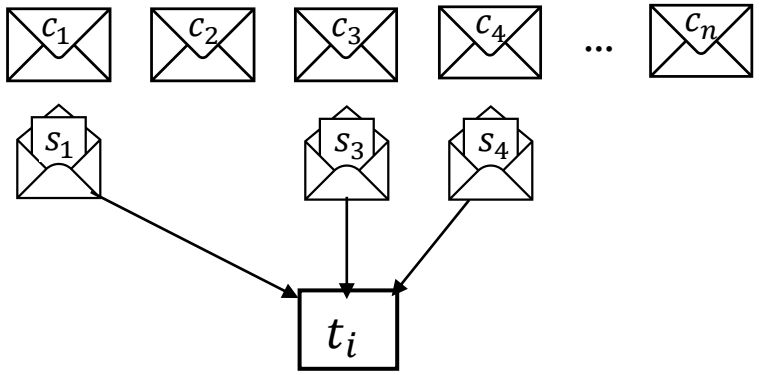
break t into blocks: $B_1 = t_1 \ldots t_B$ … B_m

Ext(B_1) = hidden bits r_1 … r_m = Ext(B_m)

Prove(I):



Proving BARG statement $i \in [mB]$:



$\pi = (\pi_{BARG}(I), B_I, c_1, \ldots, c_n)$

Note: full details are in the paper

Hiding: No PPT verifier can distinguish $r_{\bar{I}}$ from a uniform bitstring given (crs, I, r_I, π) sampled honestly

Proof: Hiding

Setup($1^\lambda, 1^m$)

crs = (crsBARG, crsHIDE)

Prover

GenBits(crs; ρ):

random seed

PRG(s_1 s_2 s_3 s_4 ... s_n)

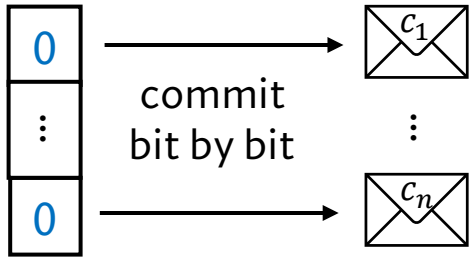
$t =$ t_1 t_2 t_3 ... t_{mB}

break t into blocks: $B_1 = t_1 \dots t_B$ ... B_m

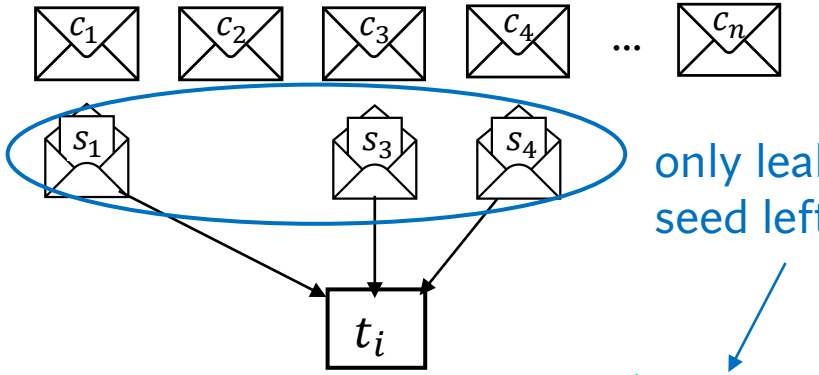
Ext(B_1) = ... Ext(B_m) =

hidden bits r_1 ... r_m

Prove(I):



Proving BARG statement $i \in [mB]$:



only leakage on PRG seed left is π_{BARG}

$\pi = (\pi_{BARG}(I), B_I, c_1, \dots, c_n)$

Note: full details are in the paper

Proof: Hiding

Hiding: No PPT verifier can distinguish $r_{\bar{I}}$ from a uniform bitstring given (crs, I, r_I, π) sampled honestly

Setup$(1^\lambda, 1^m)$

crs = (crsBARG, crsHIDE)

Prover

GenBits(crs; ρ):

by complexity leveraging and locality, t is close to uniform as long as π_{BARG} becomes correlated with t

uniform string

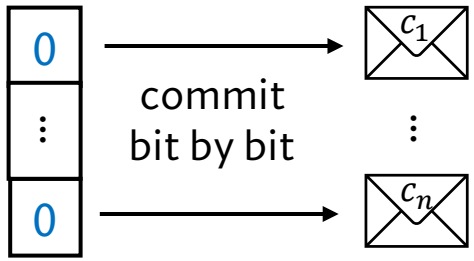
$t =$ | t_1 | t_2 | t_3 | ... | t_{mB} |

break t into blocks: | $B_1 = t_1 \dots t_B$ | ... | B_m |

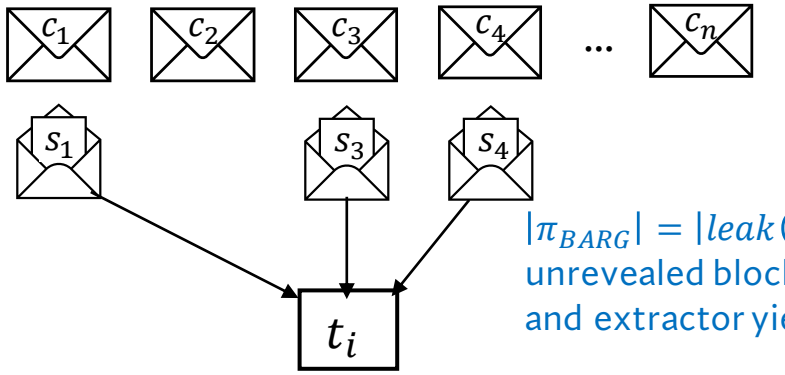
Ext(B_1) = Ext(B_m) =

hidden bits | r_1 | ... | r_m |

Prove(I):



Proving BARG statement $i \in [mB]$:



$|\pi_{BARG}| = |leak(t)| < B$, so each unrevealed block has desired entropy and extractor yields uniform $r_{\bar{I}}$

$\pi = (leak(t), B_I, c_1, \dots, c_n)$

Note: full details are in the paper

Recap: this work

Setup($1^\lambda, 1^m$)

crs = (crsBARG, crsBIND)

Note: extractor seeds also in crs

Prover

GenBits(crs; ρ):

random seed

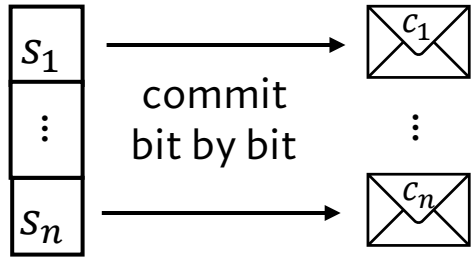
PRG(s_1 s_2 s_3 s_4 ... s_n)

$t =$ t_1 t_2 t_3 ... t_{mB}

break t into blocks: $B_1 = t_1 \dots t_B$ ... B_m

Ext(B_1) = hidden bits r_1 ... r_m = Ext(B_m)

Prove(I):

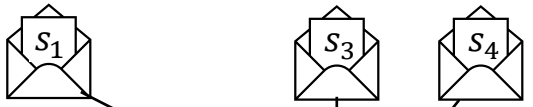


Verifier

Verify(crs, I, r_I, π): check extraction and BARG proof

Proving BARG statement $i \in [mB]$:

consider local indices



open commitments

check PRG output matches

t_i

Up to $O(\log(n))$ locality

$\pi = (\pi_{BARG}(I), B_I, c_1, \dots, c_n)$

We show: local PRG + BARG for NP + lossy PKE ⇒ NIZK for NP

Subsequent result [BKPRV23]

Focuses on understanding how batch arguments achieve statistical WI

(Somewhere sound) BARG for NP

+

Lossy PKE (dual-mode commitment)

$\Rightarrow$

Dual-mode NIZK arg. for NP with
- Computational ZK
- Adaptive soundness

or
- Statistical ZK
- Non-adaptive soundness

Open questions

- Can we weaken the lossy PKE assumption?
- Can we get ZAPs or NIWIs with similar assumptions?
- What else can we construct from BARGs?
- What other succinctness ⇒ hiding/privacy statements can we show?

Thanks for listening!

https://eprint.iacr.org/2023/695